

Proximal Algorithms and Temporal Differences for Large Linear Systems: Extrapolation, Approximation, and Simulation

Dimitri P. Bertsekas [†]

Abstract

In this paper we consider large linear fixed point problems and solution with proximal algorithms. We show that, under certain assumptions, there is a close connection between proximal iterations, which are prominent in numerical analysis and optimization, and multistep methods of the temporal difference type such as TD(λ), LSTD(λ), and LSPE(λ), which are central in simulation-based approximate dynamic programming. As an application of this connection, we provide a new and simple way to accelerate the standard proximal algorithm by extrapolation towards the multistep iteration, which generically has a faster convergence rate. We also use the connection with multistep methods to integrate into the proximal algorithmic context several new ideas that have emerged in the approximate dynamic programming context. In particular, we consider algorithms that project each proximal iterate onto the subspace spanned by a small number of basis functions, using low-dimensional calculations and simulation, and we discuss various algorithmic options from approximate dynamic programming.

1. INTRODUCTION

In this paper we focus on systems of linear equations of the form

$$x = Ax + b, \tag{1.1}$$

where A is an $n \times n$ matrix and b is a column vector in the n -dimensional space \mathbb{R}^n . We denote by $\sigma(M)$ the spectral radius of a square matrix M (maximum over the moduli of the eigenvalues of M), and we assume the following.

Assumption 1.1 The matrix $I - A$ is invertible and $\sigma(A) \leq 1$.

We consider the proximal algorithm, originally proposed for the solution of monotone variational inequalities by Martinet [Mar70] (see also the textbook treatments by Facchinei and Pang [FaP03], Bauschke

[†] The author is with the Dept. of Electr. Engineering and Comp. Science, and the Laboratory for Information and Decision Systems, M.I.T., Cambridge, Mass., 02139.

and Combettes [BaC11], and the author's [Ber15]). This algorithm has the form

$$x_{k+1} = P^{(c)}x_k,$$

where c is a positive scalar, and for a given $x \in \mathbb{R}^n$, $P^{(c)}x$ denotes the solution of the following equation in the vector y :

$$y = Ay + b + \frac{1}{c}(x - y).$$

Under Assumption 1.1, this equation has the unique solution

$$P^{(c)}x = \left(\frac{c+1}{c}I - A \right)^{-1} \left(b + \frac{1}{c}x \right), \quad (1.2)$$

because the matrix $\frac{c+1}{c}I - A$ is invertible, since its eigenvalues lie within the unit circle that is centered at $\frac{c+1}{c}$, so they do not include 0.

When A is symmetric, the system (1.1) is the optimality condition for the minimization

$$\min_{x \in \mathbb{R}^n} \left\{ \frac{1}{2}x'Qx - b'x \right\}, \quad (1.3)$$

where $Q = I - A$ and a prime denotes transposition. The proximal algorithm $x_{k+1} = P^{(c)}x_k$ can then be implemented through the minimization

$$x_{k+1} \in \arg \min_{x \in \mathbb{R}^n} \left\{ \frac{1}{2}x'Qx - b'x + \frac{1}{2c}\|x - x_k\|^2 \right\},$$

or

$$x_{k+1} = \left(\frac{1}{c}I + Q \right)^{-1} \left(b + \frac{1}{c}x_k \right).$$

In this case, Assumption 1.1 is equivalent to Q being positive definite, with all eigenvalues in the interval $(0, 2]$. Note, however, that for the minimization problem (1.3), the proximal algorithm is convergent for any positive semidefinite symmetric Q , as is well known. Thus Assumption 1.1 is not the most general assumption under which the proximal algorithm can be applied. Still, however, the assumption covers important types of problems, including the case where A is a contraction with respect to some norm, as well as applications in dynamic programming (DP for short), to be discussed shortly.

Let us denote by T the mapping whose fixed point we wish to find,

$$Tx = Ax + b.$$

We will denote by T^ℓ the ℓ -fold composition of T , where ℓ is a positive integer and we define addition of a finite number and an infinite number of linear operators in the standard way. We introduce the multistep mapping $T^{(\lambda)}$ given by

$$T^{(\lambda)} = (1 - \lambda) \sum_{\ell=0}^{\infty} \lambda^\ell T^{\ell+1}, \quad (1.4)$$

where λ is a scalar with $0 < \lambda < 1$. The series defining $T^{(\lambda)}$ is convergent under Assumption 1.1, as we will discuss later. The focus of this paper is the relation between the mappings $T^{(\lambda)}$ and $P^{(c)}$, and the ways in which this relation can be exploited algorithmically to compute x^* .

The mapping $T^{(\lambda)}$ has been central in the field that we will refer to as “approximate DP” (the name “reinforcement learning” is also often used in artificial intelligence, and the names “neuro-dynamic programming” and “adaptive dynamic programming” are often used in automatic control, with essentially the same meaning). We provide a summary of this context in this section, and we will give a more detailed discussion in Section 3. The mapping $T^{(\lambda)}$ is involved in methods for finding a fixed point of the mapping $\Pi T^{(\lambda)}$, where Π is either the identity or some form of projection onto a low-dimensional subspace S .[†] In the DP context, A is a substochastic matrix related to the Markov chain of a policy and the equation $x = Ax + b$ is the Bellman equation for the cost function x of the policy. Equations of the form $x = T^{(\lambda)}x$ are solved repeatedly within the policy iteration method, which generates a sequence of improved cost functions and associated policies. Equations of the form $x = \Pi T^{(\lambda)}x$ are solved within a corresponding approximate policy iteration method. Detailed accounts of the approximate DP context are given in several books, including the ones by Bertsekas and Tsitsiklis [BeT96], Sutton and Barto [SuB98], Si et al. [SBP04], Powell [Pow07], Busoniu et al. [BBD10], Szepesvari [Sze10], Bertsekas [Ber12a], Lewis and Liu [LeL13], and Vrabie, Vamvoudakis, and Lewis [VVL13]. Substantial computational experience has been accumulated with this methodology, and considerable success has been obtained (including prominent achievements with programs that play games, such as Backgammon, Go, and others, at impressive and sometimes above human level; see Tesauro [Tes94], Scherrer et al. [GMS13], [SMG15], and Silver et al. [MKS15], [SHM16]). In challenging approximate DP applications, the dimension of A is very high, the dimension of the approximation subspace S is low by comparison, and the large-scale computations involved in calculating the fixed point of $\Pi T^{(\lambda)}$ are handled by Monte-Carlo simulation schemes.

A variety of simulation-based methods, such as $TD(\lambda)$, $LSTD(\lambda)$, and $LSPE(\lambda)$, have been proposed in approximate DP. In particular, the fixed point iteration $x_{k+1} = \Pi T^{(\lambda)}x_k$ (where Π is orthogonal projection with respect to a weighted Euclidean norm) has been called $PVI(\lambda)$ in the author’s DP textbook [Ber12a] (PVI stands for Projected Value Iteration). Its simulation-based implementation is the $LSPE(\lambda)$ method ($LSPE$ stands for Least Squares Policy Evaluation) given in joint works of the author with his collaborators Ioffe, Nedić, Borkar, and Yu [BeI96], [NeB03], [BBN04], [YuB06], [Ber11c]. The simulation-based matrix inversion method that solves the fixed point equation $x = \Pi T^{(\lambda)}x$ is the $LSTD(\lambda)$ method, given by Bradtke and Barto [BrB96], and further discussed, extended, and analyzed by Boyan [Boy02], Lagoudakis and Parr [LaP03], Nedić and Bertsekas [NeB03], Bertsekas and Yu [BeY09], [YuB12], and Yu [Yu10], [Yu12] ($LSTD$ stands for Least Squares Temporal Differences). $TD(\lambda)$, proposed by Sutton [Sut88] in the approximate DP setting, is a stochastic approximation method for solving the equation $x = \Pi T^{(\lambda)}x$. It has the form

$$x_{k+1} = x_k + \gamma_k \left(\text{sample}(\Pi T^{(\lambda)}x_k) - x_k \right), \quad (1.5)$$

where $\text{sample}(\Pi T^{(\lambda)}x_k)$ is a stochastic simulation-generated sample of $\Pi T^{(\lambda)}x_k$, and γ_k is a diminishing stepsize satisfying standard conditions for stochastic iterative algorithms, such as $\gamma_k = 1/(k+1)$ [the paper by Tsitsiklis and VanRoy [TsV97], and the book [BeT96] provide a convergence analysis, while Bertsekas and Yu [BeY09] (Section 5.3) generalize $TD(\lambda)$ to the linear system context of this paper]. The computation of the samples in Eq. (1.5) involves simulation using Markov chains and the notion of temporal differences, which originated in reinforcement learning with the works of Samuel [Sam59], [Sam67] on a checkers-playing

[†] In approximate DP it is common to replace a fixed point equation of the form $x = F(x)$ with the equation $x = \Pi(F(x))$. This approach comes under the general framework of Galerkin approximation, which is widely used in a variety of numerical computation contexts (see e.g., the books by Krasnoselskii [Kra72] and Fletcher [Fle84], and the DP-oriented discussion in the paper [YuB10]). A distinguishing feature of approximate DP applications is that the equation $x = \Pi(F(x))$ is typically solved by simulation-based methods.

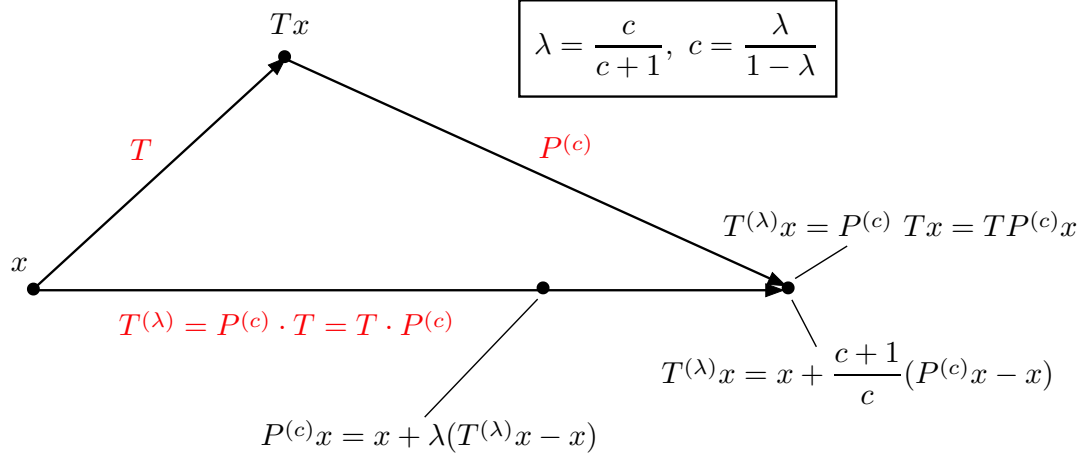


Figure 1.1. Relation of the mappings $P^{(c)}$ and $T^{(\lambda)}$. The mapping $P^{(c)}$ is obtained by interpolation between $T^{(\lambda)}$ and the identity. Reversely, $T^{(\lambda)}$ is an extrapolated form of $P^{(c)}$.

program. Mathematically, temporal differences are residual-type terms of the form $A^\ell(Ax + b - x)$, $\ell \geq 0$ (see Section 3.3). We refer to the sources given above for methods to generate samples of temporal differences in the DP/policy evaluation context, and to [BeY09] for corresponding methods and analysis in a more general linear fixed point context.

A central observation of this paper, shown in Section 2, is that the proximal mapping $P^{(c)}$ is closely related to the multistep mapping $T^{(\lambda)}$, where

$$\lambda = \frac{c}{c+1}.$$

In particular $P^{(c)}$ is an interpolated mapping between $T^{(\lambda)}$ and the identity, or reversely, $T^{(\lambda)}$ is an extrapolated form of $P^{(c)}$; see Fig. 1.1. Moreover, we show in Section 2 that under Assumption 1.1, $T^{(\lambda)}$ has a smaller spectral radius than $P^{(c)}$, and as a result extrapolation of the proximal iterates by a factor $\frac{c+1}{c}$ results in convergence acceleration. We also characterize the region of extrapolation factors that lead to acceleration of convergence, and show that it is an interval that contains $(1, 1 + 1/c]$, but may potentially be substantially larger. These facts are new to the author’s knowledge, and they are somewhat unexpected as they do not seem to readily admit an intuitive explanation.

Aside from its conceptual value and its extrapolation potential, the relation between $P^{(c)}$ and $T^{(\lambda)}$ suggests the possibility of new algorithmic approaches for large scale applications where the proximal algorithm can be used conveniently. In particular, one may consider the projected proximal algorithm,

$$x_{k+1} = \Pi P^{(c)}x_k,$$

which aims to converge to a fixed point of $\Pi P^{(c)}$. The algorithm may be based on simulation-based computations of $\Pi T^{(\lambda)}x$, and such computations have been discussed in the approximate DP context as part of the LSPE(λ) method (noted earlier), and the λ -policy iteration method (proposed in [BeI96], and further developed in the book [BeT96], and the papers [Ber12b] and [Sch13]). The simulation-based methods for computing $\Pi T^{(\lambda)}x$ has been adapted to the more general linear equation context in [BeY09]; see also [Ber12a], Section 7.3. Another possibility is to use simulation-based matrix inversion to solve the fixed point equation $x = \Pi T^{(\lambda)}x$. In the approximate DP context this is the LSTD(λ) method, which has also been extended to the general linear equations context in [BeY09].

In Section 3 of this paper, we selectively summarize without much analysis how to adapt and transfer algorithms between the TD/approximate DP and the proximal contexts. Our aim is to highlight the algorithmic possibilities that may allow us to benefit from the accumulated implementation experience within these contexts. To this end, we will draw principally on the individual and joint works of the author, M. Wang, and H. Yu; see [BeY07], [BeY09], [YuB10], [Yu10], [Yu12], [Ber11a], [Ber11b], [YuB12], [WaB13], [WaB14], and the textbook account of [Ber12a], Section 7.3, where extensions and analysis of $TD(\lambda)$, $LSTD(\lambda)$, and $LSPE(\lambda)$ for solution of the general linear system $x = \Pi T^{(\lambda)} x$ were given. This includes criteria for $T^{(\lambda)}$ and $\Pi T^{(\lambda)}$ to be a contraction, error bounds, simulation-based implementations, algorithmic variations, dealing with singularity or near singularity of $\Pi P^{(c)}$, etc.

Finally, in Section 4 we provide an extension of our acceleration result of Section 2 to nonlinear fixed point problems. In particular, we show that an extrapolated form of the proximal algorithm provides increased reduction of the distance to the fixed point over the standard proximal algorithm, provided the fixed point problem has a unique solution and involves a nonexpansive mapping (cf. Assumption 1.1). To our knowledge, this is the first simple extension of the proximal algorithm for a major class of nonlinear problems, which guarantees acceleration. On the other hand we note that the convergence theory of temporal difference methods is restricted to linear systems that satisfy Assumption 1.1. Thus a strong connection with proximal algorithms for nonlinear fixed point problems seems unlikely.

2. INTERPOLATION AND EXTRAPOLATION FORMULAS

We first review a known result from [BeY09] regarding the multistep mapping $T^{(\lambda)}$. By repeatedly applying the formula $x = Ax + b$, we can verify that

$$T^{(\lambda)}x = A^{(\lambda)}x + b^{(\lambda)}, \quad (2.1)$$

where

$$A^{(\lambda)} = (1 - \lambda) \sum_{\ell=0}^{\infty} \lambda^{\ell} A^{\ell+1}, \quad b^{(\lambda)} = \sum_{\ell=0}^{\infty} \lambda^{\ell} A^{\ell} b, \quad (2.2)$$

assuming that the series above are convergent. The following proposition shows that under Assumption 1.1, $T^{(\lambda)}$ is well defined by the power series $(1 - \lambda) \sum_{\ell=0}^{\infty} \lambda^{\ell} T^{\ell+1}$ [cf. Eq. (1.4)], and that it is a contraction with respect to some norm.

Proposition 2.1: Let Assumption 1.1 hold, and let $\lambda \in (0, 1)$.

- (a) The matrix $A^{(\lambda)}$ and the vector $b^{(\lambda)}$ are well-defined in the sense that the series in Eq. (2.2) are convergent.
- (b) The eigenvalues of $A^{(\lambda)}$ have the form

$$\theta_i = (1 - \lambda) \sum_{\ell=0}^{\infty} \lambda^{\ell} \zeta_i^{\ell+1} = \frac{\zeta_i(1 - \lambda)}{1 - \zeta_i \lambda}, \quad i = 1, \dots, n, \quad (2.3)$$

where ζ_i , $i = 1, \dots, n$, are the eigenvalues of A . Furthermore, we have $\sigma(A^{(\lambda)}) < 1$ and $\lim_{\lambda \rightarrow 1} \sigma(A^{(\lambda)}) = 0$.

The relation between the proximal mapping $P^{(c)}$ and the multistep mapping $T^{(\lambda)}$ is established in the following proposition, which is illustrated in Fig. 1.1.

Proposition 2.2: Let Assumption 1.1 hold, and let $c > 0$ and $\lambda = \frac{c}{c+1}$. Then:

(a) $P^{(c)}$ is given by

$$P^{(c)} = (1 - \lambda) \sum_{\ell=0}^{\infty} \lambda^{\ell} T^{\ell}, \quad (2.4)$$

and can be written as

$$P^{(c)}x = \overline{A}^{(\lambda)}x + \overline{b}^{(\lambda)}, \quad x \in \mathfrak{R}^n, \quad (2.5)$$

where

$$\overline{A}^{(\lambda)} = (1 - \lambda) \sum_{\ell=0}^{\infty} \lambda^{\ell} A^{\ell}, \quad \overline{b}^{(\lambda)} = \sum_{\ell=0}^{\infty} \lambda^{\ell+1} A^{\ell} b. \quad (2.6)$$

(b) We have

$$T^{(\lambda)} = TP^{(c)} = P^{(c)}T, \quad (2.7)$$

and for all $x \in \mathfrak{R}^n$,

$$P^{(c)}x = (1 - \lambda)x + \lambda T^{(\lambda)}x, \quad T^{(\lambda)}x = -\frac{1}{c}x + \frac{c+1}{c}P^{(c)}x, \quad (2.8)$$

or equivalently

$$P^{(c)}x = x + \lambda(T^{(\lambda)}x - x), \quad T^{(\lambda)}x = x + \frac{c+1}{c}(P^{(c)}x - x). \quad (2.9)$$

Proof: (a) The inverse in the definition of $P^{(c)}$ [cf. Eq. (1.2)] is written as

$$\left(\frac{c+1}{c}I - A \right)^{-1} = \left(\frac{1}{\lambda}I - A \right)^{-1} = \lambda(I - \lambda A)^{-1} = \lambda \sum_{\ell=0}^{\infty} (\lambda A)^{\ell},$$

where the power series above is convergent by Prop. 2.1(a). Thus, from Eq. (1.2) and the equation $\frac{1}{c} = \frac{1-\lambda}{\lambda}$,

$$P^{(c)}x = \left(\frac{c+1}{c}I - A \right)^{-1} \left(b + \frac{1}{c}x \right) = \lambda \sum_{\ell=0}^{\infty} (\lambda A)^{\ell} \left(b + \frac{1-\lambda}{\lambda}x \right) = (1 - \lambda) \sum_{\ell=0}^{\infty} (\lambda A)^{\ell} x + \lambda \sum_{\ell=0}^{\infty} (\lambda A)^{\ell} b,$$

which from Eq. (2.6), is equal to $\overline{A}^{(\lambda)}x + \overline{b}^{(\lambda)}$, thus proving Eq. (2.5).

(b) We have, using Eqs. (2.1), (2.2), (2.5) and (2.6),

$$TP^{(c)}x = A(\overline{A}^{(\lambda)}x + \overline{b}^{(\lambda)}) + b = (1 - \lambda) \sum_{\ell=0}^{\infty} \lambda^{\ell} A^{\ell+1}x + \sum_{\ell=0}^{\infty} \lambda^{\ell+1} A^{\ell+1}b + b = A^{(\lambda)}x + b^{(\lambda)} = T^{(\lambda)}x,$$

thus proving the left side of Eq. (2.7). The right side is proved similarly. The interpolation/extrapolation formulas (2.8) and (2.9) follow by a straightforward calculation from Eq. (2.4) and the definition $T^{(\lambda)} = (1 - \lambda) \sum_{\ell=0}^{\infty} \lambda^\ell T^{\ell+1}$ [cf. Eq. (1.4)]. As an example, the following calculation shows the left side of Eq. (2.9):

$$\begin{aligned}
x + \lambda(T^{(\lambda)}x - x) &= (1 - \lambda)x + \lambda T^{(\lambda)}x \\
&= (1 - \lambda)x + \lambda \left((1 - \lambda) \sum_{\ell=0}^{\infty} \lambda^\ell A^{\ell+1}x + \sum_{\ell=0}^{\infty} \lambda^\ell A^\ell b \right) \\
&= (1 - \lambda) \left(x + \sum_{\ell=1}^{\infty} \lambda^\ell A^\ell x \right) + \sum_{\ell=0}^{\infty} \lambda^{\ell+1} A^\ell b \\
&= \overline{A}^{(\lambda)} x + \overline{b}^{(\lambda)} \\
&= P^{(c)}x.
\end{aligned}$$

Q.E.D

We will now use the extrapolation formulas of Prop. 2.2(b) to construct variants of the proximal algorithm with interesting properties. The next proposition establishes the convergence and convergence rate properties of the proximal and multistep iterations, and shows how the proximal iteration can be accelerated by extrapolation or interpolation.

Proposition 2.3: Let Assumption 1.1 hold, and let $c > 0$ and $\lambda = \frac{c}{c+1}$. Then the eigenvalues of $\overline{A}^{(\lambda)}$ are

$$\overline{\theta}_i = \frac{1 - \lambda}{1 - \zeta_i \lambda}, \quad i = 1, \dots, n, \quad (2.10)$$

where ζ_i , $i = 1, \dots, n$, are the eigenvalues of A . Moreover, $A^{(\lambda)}$ and $\overline{A}^{(\lambda)}$ have the same eigenvectors. Furthermore, we have

$$\frac{\sigma(A^{(\lambda)})}{\sigma(A)} \leq \sigma(\overline{A}^{(\lambda)}) < 1, \quad (2.11)$$

so $\sigma(A^{(\lambda)}) < \sigma(\overline{A}^{(\lambda)})$ if $\sigma(A) < 1$.

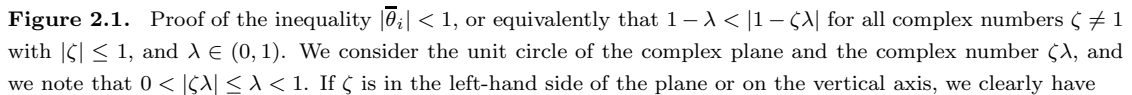
Proof: Let e_i be an eigenvector of $A^{(\lambda)}$ corresponding to the eigenvalue θ_i . By using the interpolation formula (2.8) and the eigenvalue formula (2.3) for θ_i , we have

$$\overline{A}^{(\lambda)} e_i = (1 - \lambda)e_i + \lambda A^{(\lambda)} e_i = ((1 - \lambda) + \lambda \theta_i) e_i = \left((1 - \lambda) + \lambda \frac{\zeta_i(1 - \lambda)}{1 - \zeta_i \lambda} \right) e_i = \frac{1 - \lambda}{1 - \zeta_i \lambda} e_i.$$

Hence, $\overline{\theta}_i = \frac{1 - \lambda}{1 - \zeta_i \lambda}$ and e_i are the corresponding eigenvalue and eigenvector of $\overline{A}^{(\lambda)}$, respectively.

The proof that $\sigma(\overline{A}^{(\lambda)}) < 1$, or equivalently that $|\overline{\theta}_i| < 1$ for all i , follows from a graphical argument on the complex plane, which is given in the caption of Fig. 2.1. We also have from Eqs. (2.3) and (2.10)

$$|\overline{\theta}_i| = \frac{|\theta_i|}{|\zeta_i|}, \quad i = 1, \dots, n,$$



so it is sufficient to consider the case where $\zeta \neq 0$ and the real part of ζ is positive, which is depicted in the figure. If ζ is real, we have $\zeta > 0$ as well as $\lambda > 0$, so

and we are done. If ζ is not real, we consider the isosceles triangle OAB (shaded in the figure), and note that the angles of the triangle bordering the side AB are less than 90 degrees. It follows that the angle ABC and hence also the angle ADC shown in the figure is greater than 90 degrees. Thus the side AC of the triangle ADC is strictly larger than the side DC. This is equivalent to the desired result $1 - \lambda < |1 - \zeta\lambda|$.

$$|\bar{\theta}_i| \geq \frac{|\theta_i|}{\sigma(A)}, \quad i = 1, \dots, n.$$

An interesting conclusion can be drawn from Prop. 2.3 about the convergence and the rate of convergence of the proximal iteration $x_{k+1} = P^{(c)}x_k$ and the multistep iteration $x_{k+1} = T^{(\lambda)}x_k$. Under Assumption 1.1, *both iterations are convergent, but the multistep iteration is faster when A is itself a contraction* and is

not slower otherwise; cf. Prop. 2.3(a). In the case where A is not a contraction [$\sigma(A) = 1$] it is possible that $\sigma(A^{(\lambda)}) = \sigma(\overline{A}^{(\lambda)})$ (as an example consider a case where all the eigenvalues ζ_i have modulus 1).

Even in the case where $\sigma(A^{(\lambda)}) = \sigma(\overline{A}^{(\lambda)})$, however, it is possible to accelerate the proximal iteration with an iteration that interpolates strictly between $P^{(c)}x_k$ and $T^{(\lambda)}x_k$. This is shown in the next proposition, which establishes the convergence rate properties of the extrapolated proximal iteration, and quantifies the range of extrapolation factors that lead to acceleration.

Proposition 2.4 Let Assumption 1.1 hold, and let $c > 0$ and $\lambda = \frac{c}{c+1}$. Consider any iteration that extrapolates from $P^{(c)}$ in the direction of $T^{(\lambda)}$, i.e.,

$$x_{k+1} = (1 - \gamma)P^{(c)}x_k + \gamma T^{(\lambda)}x_k, \quad \gamma > 0, \quad (2.12)$$

and write it in matrix-vector form as

$$x_{k+1} = A(\lambda, \gamma)x_k + b(\lambda, \gamma),$$

where $A(\lambda, \gamma)$ is an $n \times n$ matrix and $b(\lambda, \gamma) \in \mathbb{R}^n$. The eigenvalues of $A(\lambda, \gamma)$ are given by

$$\theta_i(\gamma) = (1 - \gamma)\overline{\theta}_i + \gamma\theta_i, \quad i = 1, \dots, n, \quad (2.13)$$

and we have

$$\sigma(A(\lambda, \gamma)) < \sigma(\overline{A}^{(\lambda)}), \quad (2.14)$$

for all γ in the interval $(0, \gamma_{\max})$, where

$$\gamma_{\max} = \max \left\{ \gamma > 0 \mid |\theta_i(\gamma)| = \overline{\theta}_i, \forall i = 1, \dots, n \right\}.$$

Moreover, we have $\gamma_{\max} \geq 1$, with equality holding if and only if $\sigma(A) = 1$.

Proof: The eigenvalue formula (2.13) follows from the interpolation formula

$$A(\lambda, \gamma) = (1 - \gamma)\overline{A}^{(\lambda)} + \gamma A^{(\lambda)},$$

and the fact that $A^{(\lambda)}$ and $\overline{A}^{(\lambda)}$ have the same eigenvectors (cf. Prop. 2.3). For each i , the scalar

$$\max \left\{ \gamma > 0 \mid |\theta_i(\gamma)| \leq |\overline{\theta}_i| \right\}$$

is the maximum extrapolation factor γ for which $\theta_i(\gamma)$ has at most as large modulus as $\overline{\theta}_i$ (cf. Fig. 2.2), and the inequality (2.14) follows. The inequality $\gamma_{\max} \geq 1$ follows from the construction of Fig. 2.2, since $|\theta_i| \leq |\overline{\theta}_i|$, $\theta_i \neq \overline{\theta}_i$, and $\gamma = 1$ corresponds to the iteration $x_{k+1} = T^{(\lambda)}x_k$. Finally, we have $\gamma_{\max} = 1$ if and only if $|\theta_i| = |\overline{\theta}_i|$ for some i , which happens if and only if $|\zeta_i| = 1$ for some i , i.e., $\sigma(A) = 1$. **Q.E.D.**

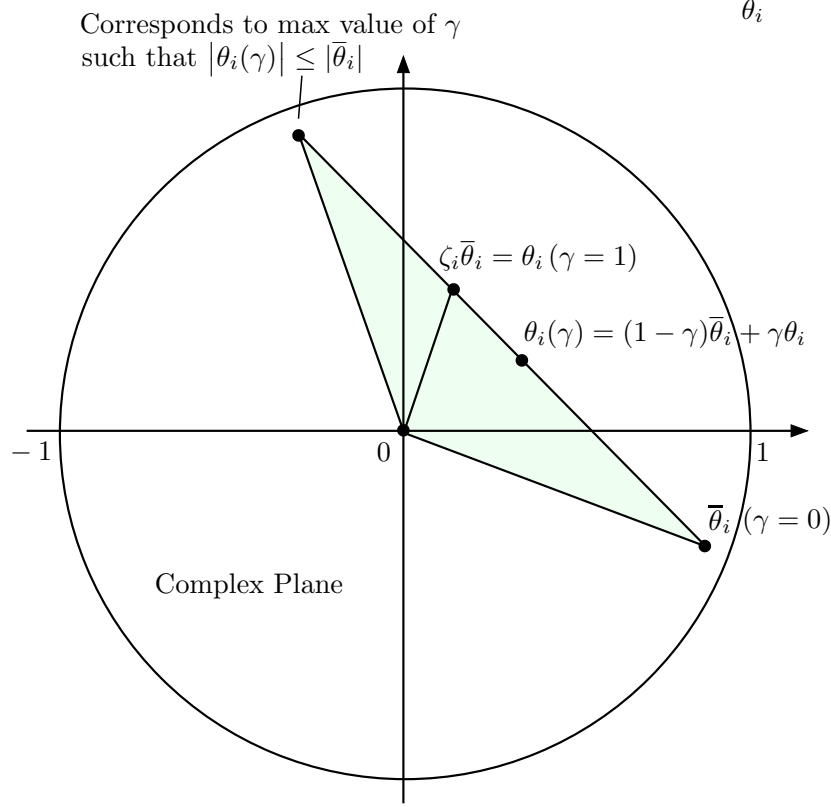


Figure 2.2. Illustration of the proof of Prop. 2.4. The eigenvalues $\theta_i(\gamma)$ of $A(\lambda, \gamma)$ are linear combinations (with $\gamma > 0$) of the eigenvalues $\bar{\theta}_i$ and $\theta_i = \zeta_i \bar{\theta}_i$ of $\bar{A}^{(\lambda)}$ and $A^{(\lambda)}$, respectively, and we have $\theta_i \leq \bar{\theta}_i$.

We may implement the extrapolation/interpolation iteration (2.12) by first implementing the proximal iteration $x_{k+1} = P^{(c)}x_k$ and then the multistep iteration according to

$$x_{k+1} = T^{(\lambda)}x_k = x_k + \frac{c+1}{c}(P^{(c)}x_k - x_k).$$

In this way, unless $\sigma(A) = 1$ [cf. Eq. (2.11)], we achieve acceleration over the proximal iteration. We may then aim for greater acceleration by extrapolating or interpolating between $P^{(c)}x_k$ and $T^{(\lambda)}x_k$ with some factor, possibly determined by experimentation [strict acceleration can always be achieved with $\gamma \in (0, 1)$]. This provides a simple and reliable method to accelerate the convergence of the proximal algorithm without knowledge of the eigenvalue structure of A beyond Assumption 1.1.[†] Conversely, we may implement the proximal iteration by interpolating the multistep iteration. Some of the possibilities along this direction will be reviewed in the next section.

[†] It is well known that when $I - A$ is a maximal monotone operator, the proximal iteration can be extrapolated by a factor of as much as two while maintaining convergence [EcB92] (see also a more refined analysis, which quantifies the effects of extrapolation, for the case where A is symmetric, given in [Ber82], Section 2.3.1). However, we are not aware of any earlier proposal of a simple and general scheme to choose an extrapolation factor that maintains convergence *and* also guarantees acceleration. Moreover, this extrapolation factor, $(c+1)/c$, may be much larger than two.

Finally, let us show that the multistep and proximal iterates $P^{(c)}x_k$ and $T^{(\lambda)}x_k$ can be computed by solving fixed point problems involving a contraction of modulus $\lambda\sigma(A)$.

Proposition 2.5 Let Assumption 1.1 hold, and let $c > 0$ and $\lambda = \frac{c}{c+1}$. The multistep and proximal iterates $T^{(\lambda)}x_k$ and $P^{(c)}x_k$ are the unique fixed points of the contraction mappings W_{x_k} and \overline{W}_{x_k} given by

$$W_{x_k}x = (1 - \lambda)Tx_k + \lambda Tx, \quad x \in \mathbb{R}^n,$$

and

$$\overline{W}_{x_k}x = (1 - \lambda)x_k + \lambda Tx, \quad x \in \mathbb{R}^n,$$

respectively.

Proof: Clearly W_{x_k} and \overline{W}_{x_k} are contraction mappings, since they are linear with spectral radius $\lambda\sigma(A) \leq \lambda < 1$. To show that $T^{(\lambda)}x_k$ is the fixed point of W_{x_k} , we must verify that $T^{(\lambda)}x_k = W_{x_k}(T^{(\lambda)}x_k)$, or equivalently that

$$T^{(\lambda)}x_k = (1 - \lambda)Tx_k + \lambda T(T^{(\lambda)}x_k) = (1 - \lambda)Tx_k + \lambda T^{(\lambda)}(Tx_k) \quad (2.15)$$

[here we are applying the formula $T(T^{(\lambda)}x) = T^{(\lambda)}(Tx)$, which is easily verified using Eqs. (2.1) and (2.2)]. In view of the interpolation formula

$$(1 - \lambda)x + \lambda T^{(\lambda)}x = P^{(c)}x, \quad \forall x \in \mathbb{R}^n, \quad (2.16)$$

[cf. Eq. (2.8)], the right-hand side of Eq. (2.15) is equal to $P^{(c)}(Tx_k)$, which from the formula $T^{(\lambda)} = P^{(c)}T$ [cf. Eq. (2.7)], is equal to $T^{(\lambda)}x_k$, the left-hand side of Eq. (2.15).

Similarly, to show that $P^{(c)}x_k$ is the fixed point of \overline{W}_{x_k} , we must verify that $P^{(c)}x_k = \overline{W}_{x_k}(P^{(c)}x_k)$, or equivalently that

$$P^{(c)}x_k = (1 - \lambda)x_k + \lambda T(P^{(c)}x_k).$$

This is proved by combining the formula $T^{(\lambda)} = TP^{(c)}$ [cf. Eq. (2.7)], and the interpolation formula (2.16). **Q.E.D.**

Proposition 2.5 suggests the iteration

$$x_{k+1} = V_m x_k, \quad (2.17)$$

where $V_m x_k$ is obtained by $m > 1$ iterations of the mapping W_{x_k} starting with x_k , i.e.,

$$V_m x_k = (W_{x_k})^m x_k,$$

so $V_m x_k$ is an approximate evaluation of $T^{(\lambda)}x_k$, the fixed point of W_{x_k} . It can be verified by induction that

$$V_m x_k = (1 - \lambda)(Tx_k + \lambda T^2 x_k + \cdots + \lambda^{m-1} T^m x_k) + \lambda^m T^m x_k,$$

and that V_m is a contraction mapping [the preceding formula is given in [BeT96], Prop. 2.7(b), while the contraction property of V_m is proved similar to Prop. 3(a) of [BeY09]]. There is also the similar iteration

$x_{k+1} = \overline{V}_m x_k$, where $\overline{V}_m x_k$ is obtained by $m > 1$ iterations of the mapping \overline{W}_{x_k} starting with x_k . This iteration may be viewed as an iterative approximate implementation of the proximal algorithm that does not require matrix inversion.

The fact that the multistep iterate $x_{k+1} = T^{(\lambda)} x_k$ is the fixed point of W_{x_k} is known in exact and approximate DP, and forms the basis for the λ -policy iteration method; see [BeI96], [BeT96], Section 2.3.1. This method admits some interesting simulation-based implementations, which have been discussed in the approximate DP literature ([Ber12b], [Sch13]), but will not be discussed further here. Based on Prop. 2.5, the proximal iteration $x_{k+1} = P^{(c)} x_k$ admits similar implementations.

3. PROJECTED PROXIMAL, PROXIMAL PROJECTED, AND TEMPORAL DIFFERENCE ALGORITHMS

In this section we aim to highlight situations where analysis and experience from approximate DP can be fruitfully transferred to the solution of linear equations by proximal-like algorithms. In particular, we discuss the simulation-based approximate solution of the equation $x = Tx$ within a subspace S spanned by a relatively small number of basis functions $\phi_1, \dots, \phi_s \in \mathbb{R}^n$. We denote by Φ the $n \times s$ matrix whose columns are ϕ_1, \dots, ϕ_s , so S can be represented as

$$S = \{\Phi r \mid r \in \mathbb{R}^s\}.$$

Instead of solving $x = Tx$ or the multistep system $x = T^{(\lambda)} x$ (which has the same solution as $x = Tx$) we consider the projected form

$$x = \Pi T^{(\lambda)} x,$$

where the mapping $\Pi : \mathbb{R}^n \mapsto \mathbb{R}^n$ is some form of projection onto S , in the sense that Π is linear, $\Pi x \in S$ for all $x \in \mathbb{R}^n$, and $\Pi x = x$ for all $x \in S$. Note that while in DP the matrix A is assumed substochastic, the methodology described in this section assumes only Assumption 1.1.

A general form of such Π is the oblique projection

$$\Pi = \Phi(\Psi' \Xi \Phi)^{-1} \Psi' \Xi, \tag{3.1}$$

where Ξ is a diagonal $n \times n$ positive semidefinite matrix with components ξ_1, \dots, ξ_n along the diagonal, and Ψ is an $n \times s$ matrix such that $\Psi' \Xi \Phi$ is invertible. Most often in approximate DP applications, the projection is orthogonal with respect to a Euclidean norm [cf. case (1)] below. However, there are situations where a more general type of projection is interesting [see cases (2) and (3) below]. The use of oblique (as opposed to orthogonal) projections in approximate DP was suggested by Scherrer [Sch10]. We note some special cases that have received attention in the approximate DP literature:

- (1) *Orthogonal projection*: Here $\Psi = \Phi$ and Ξ is positive definite. Then Π is the orthogonal projection onto S with respect to the norm corresponding to Ξ , i.e., $\|x\|^2 = \sum_{i=1}^n \xi_i x_i^2$. Solving the projected system $x = \Pi T^{(\lambda)} x$ corresponds to a form of Galerkin approximation, as noted earlier.[†]

[†] An important fact here is that there exists a (weighted) orthogonal projection Π such that $\sigma(\Pi A) \leq \sigma(A)$. Then if $T^{(\lambda)}$ or $P^{(c)}$ is a contraction the same is true for $\Pi T^{(\lambda)}$ or $\Pi P^{(c)}$, respectively (see [BeY09] for a more detailed discussion of this issue). While finding Π may not be simple in general, in approximate DP, a projection Π such that $\sigma(\Pi A) \leq \sigma(A)$ is implicitly known in terms of the stationary distribution of a corresponding Markov chain (see e.g., [BeT96], Lemma 6.4, or [Ber12], Lemma 6.3.1).

- (2) *Seminorm projection*: Here $\Psi = \Phi$, the matrix $\Phi'\Xi\Phi$ is invertible, but Ξ is only positive semidefinite (so some of the components ξ_i may be zero). Then Π is a seminorm projection with respect to the seminorm defined by Ξ . The seminorm projection Πx can be computed as the unique solution of the least squares problem

$$\min_{r \in \mathbb{R}^s} \sum_{i=1}^n \xi_i (x_i - \phi'_i r)^2, \quad (3.2)$$

where ϕ'_1, \dots, ϕ'_n are the rows of Φ (the solution is unique in view of the assumed invertibility of $\Phi'\Xi\Phi$). This type of least squares problem arises when we are trying to approximate a high dimensional vector x onto S but we know only some of the components of x (but enough to ensure that the preceding minimization has a unique solution). Seminorm projections were first considered in the approximate DP context in the paper [YuB12], and we refer to that reference for more discussion on their applications.

- (3) *Aggregation*: Here $\Pi = \Phi D$, where D is an $s \times n$ matrix, and we assume that the rows of Φ and D are probability distributions, and that Φ and D have a full rank s . We replace the solution of the system $x = Ax + b$ with the projected system $x = \Pi(Ax + b)$, or $\Phi r = \Phi D(A\Phi r + b)$, which equivalently (since ϕ has full rank) can be written as

$$r = DA\Phi r + Db.$$

This system is obtained by forming convex combinations of rows and columns of A to construct the “aggregate” matrix $DA\Phi$. Aggregation has a long history in numerical computation, and it is discussed in detail in the context of approximate DP in [Ber11b] and [Ber12a] (Sections 6.5 and 7.3.7), and the references quoted there. It turns out that for a very broad class of aggregation methods, called “aggregation with representative features” ([Ber12a], Example 6.5.4, and Exercise 7.11), the matrix ΦD is a seminorm projection, as first shown in [YuB12]. The matrix ΦD can also be viewed as an oblique projection in this case (see [Ber12a], Section 7.3.6).

While x^* is the unique solution of $x = T^{(\lambda)}x$ for all λ , the solution of the projected equation $x = \Pi T^{(\lambda)}x$ depends on λ . Also, because of the linearity of Π and the extrapolation property (2.8) shown in the preceding section, the projected proximal equation $x = \Pi P^{(c)}x$ has the same solution as the system $x = \Pi T^{(\lambda)}x$ where $\lambda = \frac{c}{c+1}$. Let us denote by x_λ this solution. Generally, for any norm $\|\cdot\|$ we have the error bound

$$\|x^* - x_\lambda\| \leq \|(I - \Pi A^{(\lambda)})^{-1}\| \|x^* - \Pi x^*\|, \quad (3.3)$$

which is derived from the following calculation:

$$x^* - x_\lambda = x^* - \Pi x^* + \Pi x^* - x_\lambda = x^* - \Pi x^* + \Pi T x^* - \Pi T^{(\lambda)} x_\lambda = x^* - \Pi x^* + \Pi A^{(\lambda)}(x^* - x_\lambda).$$

Thus the approximation error $\|x^* - x_\lambda\|$ is proportional to the “distance” $\|x^* - \Pi x^*\|$ of the solution x^* from the approximation subspace. It is well known that for values of λ close to 1, x_λ tends to approximate better the projection Πx^* . However, values of λ close to 1 correspond to large values of c , resulting in a less-well conditioned projected proximal equation $x = \Pi P^{(c)}x$. There is also a related tradeoff that is well-documented in the DP literature: as λ is increased towards 1, solving the projected equation $x = \Pi T^{(\lambda)}x$ by simulation is more time-consuming because of increased simulation noise and an associated need for more simulation samples. For further discussion of the choice of λ , we refer to the references cited earlier, and for error bounds that are related but sharper than Eq. (3.3), we refer to Yu and Bertsekas [YuB10], and Scherrer [Sch10], [Sch13].

For the case of the projection formula (3.1) and assuming that Φ has full rank, the solution x_λ of the system

$$x = \Pi T^{(\lambda)} x = \Pi(A^{(\lambda)}x + b) = \Phi(\Psi' \Xi \Phi)^{-1} \Psi' \Xi (A^{(\lambda)}x + b^{(\lambda)}),$$

can be written as

$$x_\lambda = \Phi r_\lambda,$$

where r_λ is the unique solution of the low-dimensional system of equations

$$r = (\Psi' \Xi \Phi)^{-1} \Psi' \Xi (A^{(\lambda)} \Phi r + b^{(\lambda)}).$$

Equivalently, this system is written as

$$r = Q^{(\lambda)} r + d^{(\lambda)}, \quad (3.4)$$

where

$$Q^{(\lambda)} = (\Psi' \Xi \Phi)^{-1} \Psi' \Xi A^{(\lambda)} \Phi, \quad d^{(\lambda)} = (\Psi' \Xi \Phi)^{-1} \Psi' \Xi b^{(\lambda)}. \quad (3.5)$$

By defining

$$C^{(\lambda)} = I - Q^{(\lambda)}, \quad (3.6)$$

this system can also be written as

$$C^{(\lambda)} r = d^{(\lambda)}, \quad (3.7)$$

where

$$C^{(\lambda)} = (\Psi' \Xi \Phi)^{-1} \Psi' \Xi (I - A^{(\lambda)}) \Phi, \quad d^{(\lambda)} = (\Psi' \Xi \Phi)^{-1} \Psi' \Xi b^{(\lambda)}. \quad (3.8)$$

3.1 Projected Proximal and Proximal Projected Algorithms

Let us now consider the solution of the projected equation $x = \Pi T^{(\lambda)} x$, and the equivalent systems $r = Q^{(\lambda)} r + d^{(\lambda)}$ [cf. Eq. (3.4)] and $C^{(\lambda)} r = d^{(\lambda)}$ [cf. Eq. (3.7)] with proximal-type algorithms, assuming that Φ has full rank. There are two different approaches here (which coincide when there is no projection, i.e., $S = \mathbb{R}^n$ and $\Pi = I$):

(a) Use the algorithm

$$x_{k+1} = \Pi T^{(\lambda)} x_k, \quad (3.9)$$

or equivalently [cf. Eqs. (3.4) and (3.6)],

$$r_{k+1} = Q^{(\lambda)} r_k + d^{(\lambda)} = r_k - (C^{(\lambda)} r_k - d^{(\lambda)}). \quad (3.10)$$

Another possibility is to use the interpolated version, which is the projected proximal algorithm

$$x_{k+1} = \Pi P^{(c)} x_k, \quad (3.11)$$

where $c = \frac{\lambda}{1-\lambda}$ (cf. Fig. 3.1), or equivalently, based on the interpolation formula (2.9),

$$r_{k+1} = r_k + \lambda(Q^{(\lambda)} r_k + d^{(\lambda)} - r_k) = r_k - \lambda(C^{(\lambda)} r_k - d^{(\lambda)}). \quad (3.12)$$

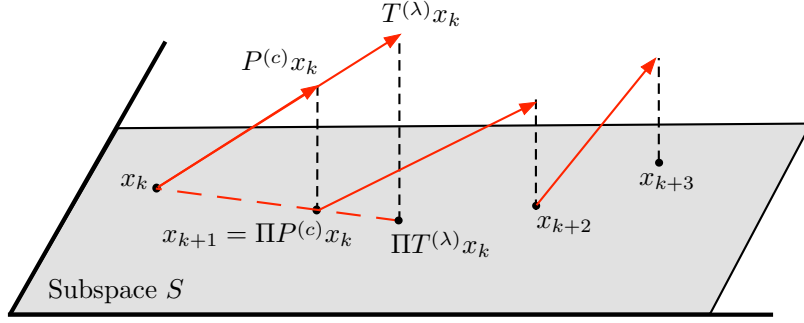


Figure 3.1. Illustration of the projected proximal algorithm (3.11) in relation to the projected multistep iteration (3.9). All iterates x_k , except possibly x_0 , lie on S .

- (b) Apply the proximal algorithm (1.2) to the system $r = Q^{(\lambda)}r + d^{(\lambda)}$ or the system $C^{(\lambda)}r = d^{(\lambda)}$ [cf. Eqs. (3.4) and (3.7)]:

$$\begin{aligned}
 r_{k+1} &= \left(\frac{\hat{c}+1}{\hat{c}} I - Q^{(\lambda)} \right)^{-1} \left(d^{(\lambda)} + \frac{1}{\hat{c}} r_k \right) \\
 &= \left(\frac{1}{\hat{c}} I + C^{(\lambda)} \right)^{-1} \left(d^{(\lambda)} + \frac{1}{\hat{c}} r_k \right) \\
 &= r_k - \left(\frac{1}{\hat{c}} I + C^{(\lambda)} \right)^{-1} (C^{(\lambda)} r_k - d^{(\lambda)}),
 \end{aligned} \tag{3.13}$$

where \hat{c} is a positive parameter that need not be related to λ [cf. Eq. (1.2)]. We may also use the extrapolated version that was discussed in the preceding section:

$$r_{k+1} = r_k - \frac{\hat{c}+1}{\hat{c}} \left(\frac{1}{\hat{c}} I + C^{(\lambda)} \right)^{-1} (C^{(\lambda)} r_k - d^{(\lambda)}), \tag{3.14}$$

(cf. Fig. 1.1). Extrapolation factors that are intermediate between 1 and $\frac{\hat{c}+1}{\hat{c}}$ or larger than $\frac{\hat{c}+1}{\hat{c}}$ may be considered. Note that this is a *two-parameter algorithm*: the parameter \hat{c} is used for regularization, and may be different from $\frac{1-\lambda}{\lambda}$. This allows some flexibility of implementation: the choice of λ should aim to strike a balance between small approximation error $\|x_\lambda - \Pi x^*\|$ and implementation difficulties due to ill-conditioning and/or simulation overhead, while the choice of \hat{c} should aim at guarding against near singularity of $C^{(\lambda)}$. For the extrapolated algorithm (3.14) to be valid, not only should $C^{(\lambda)}$ be invertible, but we must also have $\sigma(I - C^{(\lambda)}) \leq 1$. This is true under mild conditions; see [BeY09], Prop. 5.

The algorithms in (a) and (b) above are related but different. The algorithms in (a) are *projected proximal* algorithms (possibly with extrapolation), like the ones in Fig. 3.1. The algorithms in (b), use the projection and proximal operations in reverse order: they are *proximal projected* algorithms, i.e., proximal algorithms applied to the projected equation. Both types of algorithms have the generic form

$$r_{k+1} = r_k - \gamma G(C^{(\lambda)} r_k - d^{(\lambda)}),$$

where γ is a nonnegative stepsize and G is a matrix such that $GC^{(\lambda)}$ has eigenvalues with positive real parts. This algorithm and its simulation-based implementations, for both cases where $C^{(\lambda)}$ is nonsingular and singular, has been studied in detail in references [WaB13] and [WaB14]. Its convergence properties have been analyzed in these references under the assumption that the stepsize γ is sufficiently small. The

algorithms (3.10) and (3.13) have been explicitly mentioned earlier. The accelerated proximal projected algorithm (3.14) is new.

Note that the algorithms (3.13) and (3.14) make sense also when $\lambda = 0$, in which case

$$C^{(0)} = (\Psi' \Xi \Phi)^{-1} \Psi' \Xi (I - A) \Phi, \quad d^{(0)} = (\Psi' \Xi \Phi)^{-1} \Psi' \Xi b,$$

[cf. Eq. (3.8)]. Then, the algorithm (3.13) (which is known in approximate DP) can be viewed as the proximal algorithm applied to the projected system $x = \Pi T x$, while the algorithm (3.14) can be viewed as the corresponding faster multistep algorithm

$$r_{k+1} = r_k - \frac{\hat{c} + 1}{\hat{c}} \left(\frac{1}{\hat{c}} I + C^{(0)} \right)^{-1} (C^{(0)} r_k - d^{(0)}),$$

which has not been considered earlier.

3.2 Simulation-Based Methods

The difficulty with the preceding algorithms (3.10)-(3.14) is that when n is very large, the computation of $C^{(\lambda)}$ and $d^{(\lambda)}$ involves high-dimensional inner products, whose exact computation is impossible. This motivates the replacement of $C^{(\lambda)}$ and $d^{(\lambda)}$ with Monte-Carlo simulation-generated estimates, which can be computed with low-dimensional calculations. This simulation-based approach has been used and documented extensively in approximate DP since the late 80s, although the algorithms (3.12) and (3.14) are new, to our knowledge. Moreover, sampling and simulation for solution of linear systems have a long history, starting with a suggestion by von Neumann and Ulam (recounted by Forsythe and Leibler [Cur57]); see the survey by Halton [Hal70]). More recently, work on simulation methods has focused on using low-order calculations for solving large least squares and other problems. In this connection we note the papers by Strohmer and Vershynin [StV9], Censor and Herman [CeH09], and Leventhal and Lewis [LeL10] on randomized versions of coordinate descent and iterated projection methods for overdetermined least squares problems, and the series of papers by Drineas, Kannan, Mahoney, Muthukrishnan, Boutsidis, and Magdon-Ismail, who consider the use of simulation methods for linear least squares problems and low-rank matrix approximation problems; see [DKM06a], [DKM06b], [DMM06], [DMM08], [DMMS11], and [BDM14].

Let us denote by $C_k^{(\lambda)}$ and $d_k^{(\lambda)}$, respectively, the simulation-based estimates to $C^{(\lambda)}$ and $d^{(\lambda)}$, which are available at iteration k . Then the multistep iteration $x_{k+1} = \Pi T^{(\lambda)} x_k$ [cf. Eq. (3.9)] can be implemented in approximate form as

$$r_{k+1} = r_k - (C_k^{(\lambda)} r_k - d_k^{(\lambda)}), \quad (3.15)$$

[cf. Eq. (3.10)]. We implicitly assume here that at each iteration k , one or more Monte-Carlo samples are collected and added to samples from preceding iterations, in order to form $C_k^{(\lambda)}$ and $d_k^{(\lambda)}$, and that the sample collection method is such that

$$\lim_{k \rightarrow \infty} C_k^{(\lambda)} = C^{(\lambda)}, \quad \lim_{k \rightarrow \infty} d_k^{(\lambda)} = d^{(\lambda)},$$

with probability 1. The iteration (3.15) is known as LSPE(λ) in approximate DP. Thus, based on the analysis of this paper, LSPE(λ) can be viewed as an extrapolated form of the projected proximal algorithm of Fig. 3.1, implemented by simulation.

A popular alternative to LSPE(λ) is the LSTD(λ) algorithm, which approximates the solution of the projected equation $C^{(\lambda)}r = d^{(\lambda)}$ with the solution of the equation

$$C_k^{(\lambda)}r = d_k^{(\lambda)} \quad (3.16)$$

that iteration (3.15) aims to solve. In LSTD(λ) this is done by simple matrix inversion, but the main computational burden is the calculation of $C_k^{(\lambda)}$ and $d_k^{(\lambda)}$. Analysis and simulation suggests that overall the LSPE(λ) and LSTD(λ) algorithms are computationally competitive with each other; see the discussions in [BBN04], [YuB06], [Ber11b], [Ber12a]. Another prominent algorithm in approximate DP is TD(λ) [cf. Eq. (1.5)], which may be viewed as a stochastic approximation method for solving $C^{(\lambda)}r = d^{(\lambda)}$. This algorithm has a long history in approximate DP, as noted earlier, and has been extended to the general linear system context in [BeY09], Section 5.3.

A major problem for the preceding simulation-based algorithms is that when $C^{(\lambda)}$ is nearly singular, $C_k^{(\lambda)}$ should be a very accurate estimate of $C^{(\lambda)}$, so that $\sigma(I - C_k^{(\lambda)}) < 1$, which is a requirement for the methods (3.16) and (3.10) to make sense. The papers [WaB13] and [WaB14] address this issue and provide stabilization schemes to improve the performance of the LSPE(λ) and LSTD(λ) methods, as well as other iterative algorithms for solving singular and near singular systems of equations by simulation. On the other hand the proximal implementations (3.13) and (3.14) are more tolerant of near-singularity of $C^{(\lambda)}$ and simulation noise. This is a generic property of the proximal algorithm and the main motivation for its use. The simulation-based version of the algorithm (3.14), is

$$r_{k+1} = r_k - \frac{\hat{c} + 1}{\hat{c}} \left(\frac{1}{\hat{c}} I + C_k^{(\lambda)} \right)^{-1} (C_k^{(\lambda)} r_k - d_k^{(\lambda)}).$$

Its use of the extrapolation factor $\frac{\hat{c}+1}{\hat{c}}$ may provide significant acceleration, particularly for small values of \hat{c} , while simultaneously guarding against near singularity of $C^{(\lambda)}$.

Let us also mention another approach of the proximal projected type, which can also be implemented by simulation. This is to convert the projected equation $C^{(\lambda)}r = d^{(\lambda)}$ to the equivalent equation

$$C^{(\lambda)'} \Sigma^{-1} C^{(\lambda)} r = C^{(\lambda)'} \Sigma^{-1} d^{(\lambda)},$$

where Σ is a symmetric positive definite matrix, and then apply the proximal algorithm to its solution. The analog of the simulation-based proximal algorithm (3.13) is

$$r_{k+1} = r_k - \left(\frac{1}{\hat{c}} I + C_k^{(\lambda)'} \Sigma^{-1} C_k^{(\lambda)} \right)^{-1} C_k^{(\lambda)'} \Sigma^{-1} (C_k^{(\lambda)} r_k - d_k^{(\lambda)}), \quad (3.17)$$

and its extrapolated version [cf. Eq. (3.14)] is

$$r_{k+1} = r_k - \frac{\hat{c} + 1}{\hat{c}} \left(\frac{1}{\hat{c}} I + C_k^{(\lambda)'} \Sigma^{-1} C_k^{(\lambda)} \right)^{-1} C_k^{(\lambda)'} \Sigma^{-1} (C_k^{(\lambda)} r_k - d_k^{(\lambda)}).$$

These algorithms are valid assuming that $C^{(\lambda)}$ is invertible, $\sigma(I - C^{(\lambda)'} \Sigma^{-1} C^{(\lambda)}) \leq 1$, and $\lim_{k \rightarrow \infty} C_k^{(\lambda)} = C^{(\lambda)}$. The algorithm (3.17) has been considered both in the approximate DP and the more general linear system context in [WaB13], [WaB14]; see also the textbook presentation of [Ber12a], Sections 7.3.8 and 7.3.9. However, remarkably, if $C^{(\lambda)}$ is singular, its iterate sequence $\{r_k\}$ may diverge as shown by Example 9 of the paper [WaB14] (although the residual sequence $\{C_k^{(\lambda)} r_k - d_k^{(\lambda)}\}$ can be shown to converge to 0 generically; cf. Prop. 9 of [WaB14]).

3.3 The Use of Temporal Differences

The preceding discussion has outlined the general ideas of simulation-based methods, but did not address specifics. Some of the most popular implementations are based on the notion of temporal differences, which are residual-like terms of the form

$$d(x, \ell) = A^\ell(Ax + b - x), \quad x \in \mathbb{R}^n, \ell = 0, 1, \dots$$

In particular, it can be shown using the definition (1.4) that

$$T^{(\lambda)}x = x + \sum_{\ell=0}^{\infty} \lambda^\ell d(x, \ell). \quad (3.18)$$

This can be verified with the following calculation

$$\begin{aligned} T^{(\lambda)}x &= \sum_{\ell=0}^{\infty} (1-\lambda)\lambda^\ell (A^{\ell+1}x + A^\ell b + A^{\ell-1}b + \dots + b) \\ &= x + (1-\lambda) \sum_{\ell=0}^{\infty} \lambda^\ell \sum_{m=0}^{\ell} (A^m b + A^{m+1}x - A^m x) \\ &= x + (1-\lambda) \sum_{m=0}^{\infty} \left(\sum_{\ell=m}^{\infty} \lambda^\ell \right) (A^m b + A^{m+1}x - A^m x) \\ &= x + \sum_{m=0}^{\infty} \lambda^m (A^m b + A^{m+1}x - A^m x) \end{aligned}$$

from [BeY09], Section 5.2.

Based on the least squares implementation (3.2) and the temporal differences expression (3.18), and assuming that Φ has full rank s , the projected algorithm

$$\Phi r_{k+1} = \Pi T^{(\lambda)} \Phi r_k,$$

[cf. the LSPE(λ) algorithm] is given by

$$r_{k+1} = \arg \min_{r \in \mathbb{R}^s} \sum_{i=1}^n \xi_i \left(\phi_i' r - \phi_i' r_k - \sum_{\ell=0}^{\infty} \lambda^\ell d_i(\Phi r_k, \ell) \right),$$

where (ξ_1, \dots, ξ_n) is a probability distribution over the indices $1, \dots, n$, ϕ_1', \dots, ϕ_n' are the rows of Φ , and $d_i(\Phi r_k, \ell)$ is the i th component of the n -dimensional vector $d(\Phi r_k, \ell)$. Equivalently, this algorithm is written as

$$r_{k+1} = r_k + \left(\sum_{i=1}^n \xi_i \phi_i \phi_i' \right)^{-1} \sum_{i=1}^n \xi_i \phi_i \left(\sum_{\ell=0}^{\infty} \lambda^\ell d_i(\Phi r_k, \ell) \right). \quad (3.19)$$

In the simulation-based implementation of this iteration, the terms

$$\sum_{i=1}^n \xi_i \phi_i \phi_i'$$

and

$$\sum_{i=1}^n \xi_i \phi_i \left(\sum_{\ell=0}^{\infty} \lambda^\ell d_i(\Phi r_k, \ell) \right)$$

are viewed as expected values with respect to the probability distribution (ξ_1, \dots, ξ_n) , and are approximated by sample averages.

The samples may be collected in a variety of ways. Typically, they are obtained by simulating a suitable n -state Markov chain to produce one infinite sequence of indexes (i_0, i_1, \dots) or multiple finite sequences of indexes (i_0, i_1, \dots, i_m) , where m is an integer (m may be different for different sequences). Corresponding samples of the temporal differences are also collected during this process. The transition probabilities of the Markov chain are related to the elements of the matrix A , and are chosen in a way to preserve convergence of the iteration (3.19). The details of this, as well as the convergence analysis are complicated, and further review is beyond the scope of this paper. While the formalism of temporal differences is commonly used in practice, simulation-based algorithms may be implemented in other ways (see, e.g., [BeY09], Section 5.1, [Ber11c], [YuB12]). Moreover, similar ideas can be used in simulation-based versions of other related multistep algorithms, such as the one of Eq. (2.17) as well as analogs of the LSTD(λ) and TD(λ) methods. For more detailed discussions, we refer to the literature cited earlier.

4. EXTENSION TO NONLINEAR FIXED POINT PROBLEMS

In this section we consider the solution of the nonlinear fixed point problem

$$x = F(x), \tag{4.1}$$

where $F : \mathfrak{R}^n \mapsto \mathfrak{R}^n$ is a mapping satisfying the following assumption, which generalizes Assumption 1.1.

Assumption 4.1 The equation (4.1) has a unique solution denoted x^* , and F satisfies

$$\|F(x_1) - F(x_2)\| \leq \gamma \|x_1 - x_2\|, \quad \forall x_1, x_2 \in \mathfrak{R}^n, \tag{4.2}$$

where $\|\cdot\|$ is some Euclidean norm and γ is a positive scalar with $\gamma \leq 1$.

The proximal algorithm for this problem has the form

$$x_{k+1} = P^{(c)}(x_k),$$

where c is a positive scalar, and for a given $x \in \mathfrak{R}^n$, $P^{(c)}(x)$ denotes the solution of the following equation in the vector y :

$$y = F(y) + \frac{1}{c}(x - y). \tag{4.3}$$

Under Assumption 4.1 it can be verified that the mapping $x \mapsto x - F(x)$ satisfies the standard monotonicity assumption under which the proximal algorithm converges to x^* starting from any initial vector $x_0 \in \mathfrak{R}^n$.

In particular, denoting by $\langle \cdot, \cdot \rangle$ the inner product that defines the Euclidean norm $\|\cdot\|$, this assumption is written as

$$0 \leq \langle x_1 - x_2, x_1 - F(x_1) - x_2 + F(x_2) \rangle, \quad \forall x_1, x_2 \in \mathbb{R}^n,$$

or equivalently

$$\langle x_1 - x_2, F(x_1) - F(x_2) \rangle \leq \|x_1 - x_2\|^2, \quad \forall x_1, x_2 \in \mathbb{R}^n.$$

This relation holds since by the Cauchy-Schwarz inequality we have under Assumption 4.1,

$$\langle x_1 - x_2, F(x_1) - F(x_2) \rangle \leq \|x_1 - x_2\| \|F(x_1) - F(x_2)\| \leq \gamma \|x_1 - x_2\|^2 \leq \|x_1 - x_2\|^2.$$

We consider the following extrapolated version of the proximal algorithm:

$$x_{k+1} = E^{(c)}(x_k),$$

where

$$E^{(c)}(x) = x + \frac{c+1}{c} (P^{(c)}(x) - x). \quad (4.4)$$

When F is linear as in Section 1, this algorithm coincides to the multistep method $x_{k+1} = T^{(\lambda)}x_k$ (cf. Fig. 1.1). The following proposition shows that the extrapolated iterate distance $\|E^{(c)}(x_k) - x^*\|$ is no larger than the proximal iterate distance $\|P^{(c)}(x_k) - x^*\|$, and is smaller if F is contractive, i.e., if $\gamma < 1$.

Proposition 4.1: Let Assumption 4.1 hold and let $c > 0$. Then we have

$$\|E^{(c)}(x) - x^*\| \leq \gamma \|P^{(c)}(x) - x^*\|, \quad \forall x \in \mathbb{R}^n.$$

Proof: From Eq. (4.3) we have

$$P^{(c)}(x) + \frac{1}{c} (P^{(c)}(x) - x) = F(P^{(c)}(x)).$$

Using the form (4.4) of $E^{(c)}(x)$ and the preceding equation, we have

$$E^{(c)}(x) = x + \frac{c+1}{c} (P^{(c)}(x) - x) = P^{(c)}(x) + \frac{1}{c} (P^{(c)}(x) - x) = F(P^{(c)}(x)). \quad (4.5)$$

Hence, using the assumption (4.2),

$$\|E^{(c)}(x) - x^*\| \leq \|F(P^{(c)}(x)) - x^*\| = \|F(P^{(c)}(x)) - F(x^*)\| \leq \gamma \|P^{(c)}(x) - x^*\|.$$

Q.E.D.

Note that the equation

$$E^{(c)}(x) = F(P^{(c)}(x))$$

[cf. Eq. (4.5)] is the nonlinear generalization of the $T^{(\lambda)}(x) = T(P^{(c)}(x))$ [cf. Eq. (2.7)]. According to the preceding proposition, the extrapolated version of the proximal algorithm is preferable to the standard version, and it apparently should always be used when F is known to be contractive or even nonexpansive with respect to some Euclidean norm. The norm $\|\cdot\|$ and the modulus γ need not be known in order to apply the extrapolated algorithm, and there is negligible additional computation cost involved. However, the simulation-based temporal difference methodology may be used only in the case where F is linear.

5. CONCLUDING REMARKS

We have shown in this paper that proximal and multistep temporal difference methods for linear fixed point problems are closely related, and their implementations can benefit from each other, in both the exact and the approximate simulation-based setting. Our Assumption 1.1 is satisfied in broad classes of problems, including problems involving a contraction, and policy evaluation in exact and approximate DP. Some computational experience with large linear systems, beyond those arising in DP, will be helpful in quantifying the potential benefits of the extrapolation ideas of this paper. An interesting question is also how to extend the ideas of this paper from the linear equation context to the solution of linear variational inequalities, possibly with a large number of constraints, where both the proximal algorithm and multistep DP-type methods have been applied; see the papers [WaB15] and [Ber11a].

6. REFERENCES

- [BBD10] Busoniu, L., Babuska, R., De Schutter, B., and Ernst, D., 2010. Reinforcement Learning and Dynamic Programming Using Function Approximators, CRC Press, NY.
- [BBN04] Bertsekas, D. P., Borkar, V. S., and Nedić, A., 2004. “Improved Temporal Difference Methods with Linear Function Approximation,” in Learning and Approximate Dynamic Programming, by J. Si, A. Barto, W. Powell, and D. Wunsch (Eds.), IEEE Press, NY.
- [BDM14] Boutsidis, C., Drineas, P., and Magdon-Ismail, M., 2014. “Near-Optimal Column-Based Matrix Reconstruction,” SIAM J. on Computing, Vol. 43, pp. 687-717.
- [BaC11] Bauschke, H. H., and Combettes, P. L., 2011. Convex Analysis and Monotone Operator Theory in Hilbert Spaces, Springer, NY.
- [BeI96] Bertsekas, D. P., and Ioffe, S., 1996. “Temporal Differences-Based Policy Iteration and Applications in Neuro-Dynamic Programming,” Lab. for Info. and Decision Systems Report LIDS-P-2349, MIT.
- [BeT96] Bertsekas, D. P., and Tsitsiklis, J. N., 1996. Neuro-Dynamic Programming, Athena Scientific, Belmont, MA.
- [BeY07] Bertsekas, D. P., and Yu, H., 2007. “Solution of Large Systems of Equations Using Approximate Dynamic Programming Methods,” Lab. for Information and Decision Systems Report LIDS-P-2754, MIT.
- [BeY09] Bertsekas, D. P., and Yu, H., 2009. “Projected Equation Methods for Approximate Solution of Large Linear Systems,” Journal of Computational and Applied Mathematics, Vol. 227, pp. 27-50.
- [Ber82] Bertsekas, D. P., 1982. Constrained Optimization and Lagrange Multiplier Methods, Academic Press, NY; republished by Athena Scientific, Belmont, MA, 1997.
- [Ber11a] Bertsekas, D. P., 2011. “Temporal Difference Methods for General Projected Equations,” IEEE Trans. on Automatic Control, Vol. 56, pp. 2128 - 2139.
- [Ber11b] Bertsekas, D. P., 2011. “Approximate Policy Iteration: A Survey and Some New Methods,” J. of Control Theory and Applications, Vol. 9, 2011, pp. 310-335.
- [Ber11c] Bertsekas, D. P., 2011. “ λ -Policy Iteration: A Review and a New Implementation,” Lab. for Info. and Decision Systems Report LIDS-P-2874, MIT; appears in Reinforcement Learning and Approximate Dynamic Programming for Feedback Control, by F. Lewis and D. Liu (eds.), IEEE Press, 2012.
- [Ber12a] Bertsekas, D. P., 2012. Dynamic Programming and Optimal Control: Approximate Dynamic Programming, 4th Edition, Vol. II, Athena Scientific, Belmont, MA.
- [Ber12b] Bertsekas, D. P., 2012. “ λ -Policy Iteration: A Review and a New Implementation,” in Reinforcement Learning and Approximate Dynamic Programming for Feedback Control, by F. Lewis and D. Liu (eds.), IEEE Press, 2012.

- [Ber15] Bertsekas, D. P., 2015. *Convex Optimization Algorithms*, Athena Scientific, Belmont, MA.
- [Boy02] Boyan, J. A., 2002. “Technical Update: Least-Squares Temporal Difference Learning,” *Machine Learning*, Vol. 49, pp. 1-15.
- [BrB96] Bradtke, S. J., and Barto, A. G., 1996. “Linear Least-Squares Algorithms for Temporal Difference Learning,” *Machine Learning*, Vol. 22, pp. 33-57.
- [CeH09] Censor, J., Herman, G. T., and Jiang, M., 2009. “A Note on the Behavior of the Randomized Kaczmarz Algorithm of Strohmer and Vershynin,” *J. Fourier Analysis and Applications*, Vol. 15, pp. 431-436.
- [Cur54] Curtiss, J. H., 1954. “A Theoretical Comparison of the Efficiencies of Two Classical Methods and a Monte Carlo Method for Computing One Component of the Solution of a Set of Linear Algebraic Equations,” *Proc. Symposium on Monte Carlo Methods*, pp. 191-233.
- [Cur57] Curtiss, J. H., 1957. “A Monte Carlo Methods for the Iteration of Linear Operators,” *Uspekhi Mat. Nauk*, Vol. 12, pp. 149-174.
- [DKM06a] Drineas, P., Kannan, R., and Mahoney, M. W., 2006. “Fast Monte Carlo Algorithms for Matrices I: Approximating Matrix Multiplication,” *SIAM J. Computing*, Vol. 35, pp. 132-157.
- [DKM06b] Drineas, P., Kannan, R., and Mahoney, M. W., 2006. “Fast Monte Carlo Algorithms for Matrices II: Computing a Low-Rank Approximation to a Matrix,” *SIAM J. Computing*, Vol. 36, pp. 158-183.
- [DMM06] Drineas, P., Mahoney, M. W., and Muthukrishnan, S., 2006. “Sampling Algorithms for L2 Regression and Applications,” *Proc. 17th Annual SODA*, pp. 1127-1136.
- [DMM08] Drineas, P., Mahoney, M. W., and Muthukrishnan, S., 2008. “Relative-Error CUR Matrix Decompositions,” *SIAM J. Matrix Anal. Appl.*, Vol. 30, pp. 844-881.
- [DMM11] Drineas, P., Mahoney, M. W., Muthukrishnan, S., and Sarlos, T., 2011. “Faster Least Squares Approximation,” *Numerische Mathematik*, Vol. 117, pp. 219-249.
- [EcB92] Eckstein, J., and Bertsekas, D. P., 1992. “On the Douglas-Rachford Splitting Method and the Proximal Point Algorithm for Maximal Monotone Operators,” *Math. Programming*, Vol. 55, pp. 293-318.
- [FaP03] Facchinei, F., and Pang, J.-S., 2003. *Finite-Dimensional Variational Inequalities and Complementarity Problems*, Springer Verlag, NY
- [Fle84] Fletcher, C. A. J., 1984. *Computational Galerkin Methods*, Springer-Verlag, NY.
- [FoL50] Forsythe, G. E., and Leibler, R. A., 1950. “Matrix Inversion by a Monte Carlo Method,” *Mathematical Tables and Other Aids to Computation*, Vol. 4, pp. 127-129.
- [GMS13] Gabillon, V., Ghavamzadeh, M., and Scherrer, B., 2013. “Approximate Dynamic Programming Finally Performs Well in the Game of Tetris,” In *Advances in Neural Information Processing Systems*, pp. 1754-1762.
- [Hal70] Halton, J. H., 1970. “A Retrospective and Prospective Survey of the Monte Carlo Method,” *SIAM Review*, Vol. 12, pp. 1-63.
- [Kra72] Krasnoselskii, M. A., et. al, 1972. *Approximate Solution of Operator Equations*, Translated by D. Louvish, Wolters-Noordhoff Pub., Groningen.
- [LaP03] Lagoudakis, M. G., and Parr, R., 2003. “Least-Squares Policy Iteration,” *J. of Machine Learning Research*, Vol. 4, pp. 1107-1149.
- [LeL10] Leventhal, D., and Lewis, A. S., 2010. “Randomized Methods for Linear Constraints: Convergence Rates and Conditioning,” *Mathematics of Operations Research*, Vol. 35, pp. 641-654.
- [LeL13] Lewis, F. L., and Liu, D., (Eds), 2013. *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*, Wiley, Hoboken, N. J.
- [Mar70] Martinet, B., 1970. “Regularisation d’ Inequations Variationnelles par Approximations Successives,” *Rev. Francaise Inf. Rech. Oper.*, pp. 154-159.

- [MKS15] Mnih, V., Kavukcuoglu, K., Silver, D., et al., 2015. “Human-Level Control Through Deep Reinforcement Learning,” *Nature*, Vol. 518, pp. 529-533.
- [NeB03] Nedić, A., and Bertsekas, D. P., 2003. “Least Squares Policy Evaluation Algorithms with Linear Function Approximation,” *Discrete Event Dynamic Systems: Theory and Applications*, Vol. 13, pp. 79-110.
- [Pow07] Powell, W. B., 2007. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, Wiley, NY.
- [SBP04] Si, J., Barto, A., Powell, W., and Wunsch, D., (Eds.), 2004. *Learning and Approximate Dynamic Programming*, IEEE Press, NY.
- [SHM16] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, S., et al. 2016. “Mastering the Game of Go with Deep Neural Networks and Tree Search,” *Nature*, Vol. 529, pp. 484-489.
- [SMG15] Scherrer, B., Ghavamzadeh, M., Gabillon, V., Lesner, B., and Geist, M., 2015. “Approximate Modified Policy Iteration and its Application to the Game of Tetris,” *J. of Machine Learning Research*, Vol. 16, pp. 1629-1676.
- [Sam59] Samuel, A. L., 1959. “Some Studies in Machine Learning Using the Game of Checkers,” *IBM Journal of Research and Development*, pp. 210-229.
- [Sam67] Samuel, A. L., 1967. “Some Studies in Machine Learning Using the Game of Checkers. II – Recent Progress,” *IBM Journal of Research and Development*, pp. 601-617.
- [Sch10] Scherrer, B., 2010. “Should One Compute the Temporal Difference Fixed Point or Minimize the Bellman Residual? The Unified Oblique Projection View,” *Proc. of 2010 ICML*, Haifa, Israel.
- [Sch13] Scherrer, B., 2013. “Performance Bounds for λ -Policy Iteration and Application to the Game of Tetris,” *J. of Machine Learning Research*, Vol. 14, pp. 1181-1227.
- [StV09] Strohmer, T., and Vershynin, R., 2009. “A Randomized Kaczmarz Algorithm with Exponential Convergence”, *J. of Fourier Analysis and Applications*, Vol. 15, pp. 262-178.
- [SuB98] Sutton, R. S., and Barto, A. G., 1998. *Reinforcement Learning*, MIT Press, Cambridge, MA.
- [Sut88] Sutton, R. S., 1988. “Learning to Predict by the Methods of Temporal Differences,” *Machine Learning*, Vol. 3, pp. 9-44.
- [Sze10] Szepesvari, C., 2010. *Algorithms for Reinforcement Learning*, Morgan and Claypool Publishers.
- [Tes94] Tesauro, G. J., 1994. “TD-Gammon, a Self-Teaching Backgammon Program, Achieves Master-Level Play,” *Neural Computation*, Vol. 6, pp. 215-219.
- [ThS09] Thiery, C., and Scherrer, B., 2009. “Improvements on Learning Tetris with Cross-Entropy,” *International Computer Games Association Journal*, Vol. 32, pp. 23-33.
- [TsV97] Tsitsiklis, J. N., and Van Roy, B., 1997. “An Analysis of Temporal-Difference Learning with Function Approximation,” *IEEE Transactions on Automatic Control*, Vol. 42, pp. 674-690.
- [VVL13] Vrabie, D., Vamvoudakis, K. G., and Lewis, F. L., 2013. *Optimal Adaptive Control and Differential Games by Reinforcement Learning Principles*, The Institution of Engineering and Technology, London.
- [WaB13] Wang, M., and Bertsekas, D. P., 2013. “Stabilization of Stochastic Iterative Methods for Singular and Nearly Singular Linear Systems,” *Math. of Operations Research*, Vol. 39, pp. 1-30.
- [WaB14] Wang, M., and Bertsekas, D. P., 2014. “Convergence of Iterative Simulation-Based Methods for Singular Linear Systems,” *Stochastic Systems*, Vol. 3, pp. 39-96.
- [WaB15] Wang, M., and Bertsekas, D. P., 2015. “Incremental Constraint Projection Methods for Variational Inequalities,” *Math. Programming*, Vol. 150, pp. 321-363.
- [Was52] Wasow, W. R., 1952. “A Note on Inversion of Matrices by Random Walks,” *Mathematical Tables and Other Aids to Computation*, Vol. 6, pp. 78-81.
- [YuB06] Yu, H., and Bertsekas, D. P., 2006. “Convergence Results for Some Temporal Difference Methods Based on Least Squares,” *IEEE Trans. on Aut. Control*, Vol. 54, pp. 1515-153.

- [YuB10] Yu, H., and Bertsekas, D. P., 2010. “Error Bounds for Approximations from Projected Linear Equations,” *Math. of Operations Research*, Vol. 35, pp. 306-329.
- [YuB12] Yu, H., and Bertsekas, D. P., 2012. “Weighted Bellman Equations and their Applications in Dynamic Programming,” *Lab. for Information and Decision Systems Report LIDS-P-2876*, MIT.
- [Yu10] Yu, H., 2010. “Convergence of Least Squares Temporal Difference Methods Under General Conditions,” *Proc. of the 27th ICML*, Haifa, Israel.
- [Yu12] Yu, H., 2012. “Least Squares Temporal Difference Methods: An Analysis Under General Conditions,” *SIAM J. on Control and Optimization*, Vol. 50, pp. 3310-3343.